DW01_ash

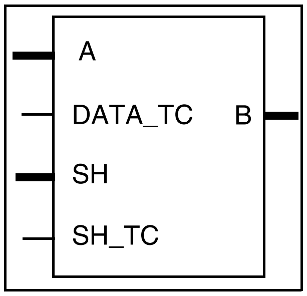
Arithmetic Shifter

Version, STAR, and myDesignWare Subscriptions: IP Directory

Features and Benefits

Revision History

- Parameterized word length
- Parameterized shift coefficient width
- Inferable using a function call



Description

DW01_ash is a general-purpose arithmetic shifter. The input data `A` is shifted left or right by the number of bits specified by the control input `SH`.

Table 1-1 Pin Description

Pin Name	Width	Direction	Function
A	*A_width* bits	Input	Input data
DATA_TC	1 bit	Input	Data two's complement control ■ 0 = Unsigned ■ 1 = Signed
SH	*SH_width* bits	Input	Shift control
SH_TC	1 bit	Input	Shift two's complement control ■ 0 = Unsigned ■ 1 = Signed
B	*A_width* bits	Output	Output data

Table 1-2 Parameter Description

Parameter	Values	Description
A_width	≥ 2	Word length of `A` and `B`
SH_width	≥ 1	Word length of `SH`

Table 1-3 Synthesis Implementations[a]

Implementation	Function	License Feature Required
mx2	Implement using 2:1 multiplexers only.	none
str	Synthesis model	DesignWare
astr	Synthesis model	DesignWare

a. During synthesis, Design Compiler selects the appropriate architecture for your constraints. However, you may force Design Compiler to use one of the architectures described in this table.

Table 1-4 Simulation Models

Model	Function
DW01.DW01_ASH_CFG_SIM	Design unit name for VHDL simulation
dw/dw01/src/DW01_ash_sim.vhd	VHDL simulation model source code
dw/sim_ver/DW01_ash.v	Verilog simulation model source code

When the control signal `SH_TC` = 0, the coefficient `SH` is interpreted as an unsigned positive number and DW01_ash performs only left shift operations.

When `SH_TC` = 1, `SH` is a two's complement number, with a negative coefficient performing a right shift and a positive coefficient performing a left shift.

The input data `A` is interpreted as an unsigned number when `DATA_TC` = 0 or a two's complement number when `DATA_TC` = 1.

The type of `A` is only significant for right shift operations, where zero padding is done on the most significant bits for unsigned data and sign extension is done for signed two's complement data.

Table 1-5 Truth Table (*SH_width* = 3, *A_width* = 8)

SH(2:0)	SH_TC	DATA_TC	B(7)	B(6)	B(5)	B(4)	B(3)	B(2)	B(1)	B(0)
000	0	X	A(7)	A(6)	A(5)	A(4)	A(3)	A(2)	A(1)	A(0)
001	0	X	A(6)	A(5)	A(4)	A(3)	A(2)	A(1)	A(0)	0
010	0	X	A(5)	A(4)	A(3)	A(2)	A(1)	A(0)	0	0
011	0	X	A(4)	A(3)	A(2)	A(1)	A(0)	0	0	0
100	0	X	A(3)	A(2)	A(1)	A(0)	0	0	0	0
101	0	X	A(2)	A(1)	A(0)	0	0	0	0	0
110	0	X	A(1)	A(0)	0	0	0	0	0	0
111	0	X	A(0)	0	0	0	0	0	0	0
000	1	X	A(7)	A(6)	A(5)	A(4)	A(3)	A(2)	A(1)	A(0)
001	1	X	A(6)	A(5)	A(4)	A(3)	A(2)	A(1)	A(0)	0
010	1	X	A(5)	A(4)	A(3)	A(2)	A(1)	A(0)	0	0
011	1	X	A(4)	A(3)	A(2)	A(1)	A(0)	0	0	0
100	1	0	0	0	0	0	A(7)	A(6)	A(5)	A(4)
101	1	0	0	0	0	A(7)	A(6)	A(5)	A(4)	A(3)
110	1	0	0	0	A(7)	A(6)	A(5)	A(4)	A(3)	A(2)
111	1	0	0	A(7)	A(6)	A(5)	A(4)	A(3)	A(2)	A(1)
100	1	1	A(7)	A(7)	A(7)	A(7)	A(7)	A(6)	A(5)	A(4)
101	1	1	A(7)	A(7)	A(7)	A(7)	A(6)	A(5)	A(4)	A(3)
110	1	1	A(7)	A(7)	A(7)	A(6)	A(5)	A(4)	A(3)	A(2)
111	1	1	A(7)	A(7)	A(6)	A(5)	A(4)	A(3)	A(2)	A(1)

Related Topics

- Math – Arithmetic Overview
- DesignWare Building Block IP User Guide

HDL Usage Through Function Inferencing - VHDL

```
library IEEE, DWARE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_arith.all;
use DWARE.DW_foundation_arith.all;

entity DW01_ash_func is
  generic(func_A_width:integer:=8; func_SH_width : integer := 3);
  port(func_A: in std_logic_vector(func_A_width-1 downto 0);
       func_DATA_TC : in std_logic;
       func_SH: in std_logic_vector(func_SH_width-1 downto 0);
       func_SH_TC : in std_logic;
       B_func: out std_logic_vector(func_A_width-1 downto 0));
end DW01_ash_func;

architecture func of DW01_ash_func is
begin
  process (func_DATA_TC, func_SH_TC, func_A,func_SH)
  begin
    if func_DATA_TC = '0' and func_SH_TC = '0' then
      B_func <= std_logic_vector(DWF_ash(unsigned(func_A),unsigned(func_SH)));
    elsif func_DATA_TC = '1' and func_SH_TC = '0' then
      B_func <= std_logic_vector(DWF_ash(signed(func_A),unsigned(func_SH)));
    elsif func_DATA_TC = '1' and func_SH_TC = '1' then
      B_func <= std_logic_vector(DWF_ash(signed(func_A),signed(func_SH)));
    else
      B_func <= std_logic_vector(DWF_ash(unsigned(func_A),signed(func_SH)));
    end if;
  end process;
end func;
```

HDL Usage Through Function Inferencing - Verilog

```
module DW01_ash_func (func_A, func_DATA_TC, func_SH, func_SH_TC, B_func);
  parameter func_A_width = 8;
  parameter func_SH_width = 3;

  // Passes the widths to the arithmetic shifter function
  parameter A_width = func_A_width;
  parameter SH_width = func_SH_width;

  // Please add search_path = search_path + {synopsys_root + "/dw/sim_ver"}
  // to your .synopsys_dc.setup file (for synthesis) and add
  // +incdir+$SYNOPSYS/dw/sim_ver+ to your verilog simulator command line
  // (for simulation).
  `include "DW01_ash_function.inc"

  input [func_A_width-1:0] func_A;
  input [func_SH_width-1:0] func_SH;
  input func_DATA_TC, func_SH_TC;
  output [func_A_width-1:0] B_func;
  reg [func_A_width-1:0] B_func;

  // infer DW01_ash
  always @ (func_A or func_DATA_TC or func_SH or func_SH_TC)
  begin
    casex({func_DATA_TC,func_SH_TC}) // synopsys full_case
      2'b00: B_func = DWF_ash_uns_uns(func_A,func_SH);
      2'b10: B_func = DWF_ash_tc_uns(func_A,func_SH);
      2'b01: B_func = DWF_ash_uns_tc(func_A,func_SH);
      2'b11: B_func = DWF_ash_tc_tc(func_A,func_SH);
    endcase
  end
endmodule
```

HDL Usage Through Component Instantiation - VHDL

```
library IEEE,DWARE;
use IEEE.std_logic_1164.all;
use DWARE.DWpackages.all;
use DWARE.DW_foundation_comp.all;

entity DW01_ash_inst is
  generic ( inst_A_width : POSITIVE := 8;
            inst_SH_width : POSITIVE := 8 );
  port ( inst_A       : in std_logic_vector(inst_A_width-1 downto 0);
         inst_DATA_TC : in std_logic;
         inst_SH      : in std_logic_vector(inst_SH_width-1 downto 0);
         inst_SH_TC   : in std_logic;
         B_inst       : out std_logic_vector(inst_A_width-1 downto 0) );
end DW01_ash_inst;

architecture inst of DW01_ash_inst is
begin

  -- Instance of DW01_ash
  U1 : DW01_ash
    generic map ( A_width => inst_A_width, SH_width => inst_SH_width )
    port map ( A => inst_A, DATA_TC => inst_DATA_TC, SH => inst_SH,
               SH_TC => inst_SH_TC, B => B_inst );
  end inst;

-- pragma translate_off
configuration DW01_ash_inst_cfg_inst of DW01_ash_inst is
  for inst
  end for; -- inst
end DW01_ash_inst_cfg_inst;
-- pragma translate_on
```

HDL Usage Through Component Instantiation - Verilog

```
module DW01_ash_inst( inst_A, inst_DATA_TC, inst_SH, inst_SH_TC, B_inst );

  parameter A_width = 8;
  parameter SH_width = 8;

  input [A_width-1 : 0] inst_A;
  input inst_DATA_TC;
  input [SH_width-1 : 0] inst_SH;
  input inst_SH_TC;
  output [A_width-1 : 0] B_inst;

  // Instance of DW01_ash
  DW01_ash #(A_width, SH_width)
    U1 (.A(inst_A), .DATA_TC(inst_DATA_TC), .SH(inst_SH),
        .SH_TC(inst_SH_TC), .B(B_inst) );

endmodule
```

Revision History

For notes about this release, see the *DesignWare Building Block IP Release Notes*.

For lists of both known and fixed issues for this component, refer to the STAR report.

For a version of this datasheet with visible change bars, click here.

Date	Release	Updates
January 2019	DWBB_201806.5	■ Updated example in "HDL Usage Through Component Instantiation - VHDL" on page 6 ■ Added this Revision History table and the document links on this page